National College of Ireland

Feature-Focused Defence: The Role of Attention Mechanisms in Enhancing Adversarial Attack Detection

MSc Research Project
Master of Science in Data Analytics

Mahesh Ryada
Student ID: 23318686

School of Computing
National College of Ireland

Supervisor: Abdul Qayum

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Mahesh Ryada

Student ID: 23318686

Programme: Master of Science in Data Analytics **Year:** 2026

Module: MSc Research Practicum Part 2

Supervisor: Abdul Qayum

Submission Due Date: 28/01/2026

Project Title: Feature-Focused Defense: The Role of Attention Mechanisms in Enhancing Adversarial Attack Detection

Word Count: 981 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Mahesh Ryada

Date: 28/01/2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Feature-Focused Defence: The Role of Attention Mechanisms in Enhancing Adversarial Attack Detection

Mahesh Ryada
X23318686

1. Introduction

The project will construct and test an entire adversarial attack detection pipeline on turbofan engine sensor data based on machine learning, deep learning, and a feature-tokenized transformer architecture. It aims at detecting the fine adversarial perturbations which are more similar to natural sensor variation, and ensuring the predictive maintenance systems to be reliable. It is not an easy task since adversarial noise takes advantage of the complexity of relationships between features and, in many cases, cannot be distinguished by typical operational behaviour. The unstable features represent a challenge to the traditional models that use static feature weighting and deep learning models that use unsteady representations. To overcome this, the system incorporates intensive pre-processing, balancing of the classes, scaling, exploratory analysis and a systematic sequence of base models and advanced models, such as the Logistic Regression, Random Forest, neural networks, and the FT-Transformer. The pipeline is amenable to complete automation of the training, tuning, validation, and evaluation process, which guarantees reproducibility and similarity in the performance evaluation of all steps.

2. System Specifications

The experiments were implemented in a controlled computational environment that was capable of supporting both deep learning workloads and transformer-based workloads. The system requirements are as given below.

Hardware Requirements

– **CPU:** Multi-core processor (minimum 4 cores; recommended 8+ cores)
– **GPU:** NVIDIA GPU with CUDA capability (recommended ≥ 6 GB VRAM)
– **RAM:** Minimum 8 GB
– **Storage:** Minimum 10 GB available space for datasets, checkpoints, and logs

Software Requirements

– **Operating System:** Windows 10/11, macOS, or Ubuntu 20.04+
– **Python Version:** Python 3.9+
– **CUDA Toolkit:** Version 11.x recommended for TensorFlow/PyTorch GPU execution
– **cuDNN:** Version compatible with installed CUDA (e.g., cuDNN 8.x)
– **Environment:** Virtual environment created via conda or venv

Deep Learning Dependencies

– TensorFlow 2.x for MLP and custom NN
– PyTorch ≥ 1.12 for FT-Transformer

3. Required Python Libraries

The system is based on the combination of scientific computing, visualization, machine learning, and deep learning libraries. Core data processing is performed with NumPy and Pandas, whereas the exploratory visualization is supported by Matplotlib and Seaborn. Scikit-learn is compatible with classical ML models, pre-processing tools, and hyperparameter optimization. The Imbalanced-learn offers SMOTE when balancing the classes. Deep and transformer-based modelling is made possible by TensorFlow/Keras and PyTorch respectively.

Installation Commands:

pip install NumPy pandas matplotlib seaborn scikit-learn imbalanced-learn
pip install TensorFlow==2.11
pip install torch torchvision torch audio --index-url
https://download.pytorch.org/whl/cu118
pip install lightgbm

4. Dataset description

The data employed in this analysis includes engine working measurements and countermeasures of adversarial conditions of predictive maintenance in aircraft settings. It contains variables like the values of operating settings, core shaft speed after correction, the variation of speed, the fuel-air qualities, the turbine bleed flow and valve status, and airflow at the station 31, and two binary labels, which are engine health condition and the adversarial manipulation status. The rows of the sample reflect the data structure and present the values of such features as Corrected_Core_Speed_RPM, Core_Speed_Variation, and Airflow_at_Station along with health labels and indicators of adversarial attacks. Every column will give information about engine behavior, degradation patterns and the effects of attack-induced perturbations. The data is full and free of redundant or missing data, and therefore is good to preprocess, analyze, and develop a model. The target variable of this paper is Adversarial_Attack_Label that states whether a sample is clean or adversaries.

5. Models Implemented

The project applies a systematic sequence of models that start with classical machine learning models, including Logistic Regression, Random Forest, and Light GBM that are used as the baseline detectors of adversarial behaviour in sensor data. Deep learning models, namely a Multi-Layer Perceptron and a home-cooked neural network, are then used which are structured to be able to capture more complex, non-linear patterns of features. Lastly, the paper uses a sophisticated attention-based FT-Transformer, which tokenizes each feature and uses multi-head self-attention to dynamically learn feature interactions. This development of basic baselines to deep architecture and transformer architecture allows an overall assessment of model performance and strength against adversarial perturbations.

Table 1. Model Performances Results

Model Name	Accuracy	Macro Averaged Precision	Macro Averaged Recall	Macro Averaged F1 Score
Logistic Regression	0.70	0.58	0.61	0.59
Random Forest	0.84	0.91	0.60	0.62
Light GBM	0.61	0.51	0.52	0.50
Multi-Layer Perceptron	0.84	0.92	0.60	0.62
Custom Neural Network	0.84	0.92	0.60	0.62
FT-Transformer	**0.93**	**0.95**	**0.82**	**0.87**

6. Code Snippets:

1) Checking for Null/missing values

```
df.isna().sum()
```

	0
Operating_Setting_2	0
Corrected_Core_Speed_RPM	0
Core_Speed_Variation	0
Fuel_Air_Ratio	0
High_Pressure_Turbine_Bleed_Flow	0
High_Pressure_Turbine_Bleed_Status	0
Airflow_at_Station_31	0
Remaining_Useful_Life_Status	0
Adversarial_Attack_Label	0

2) Data Set information

```
df.info() # Dataset information
```

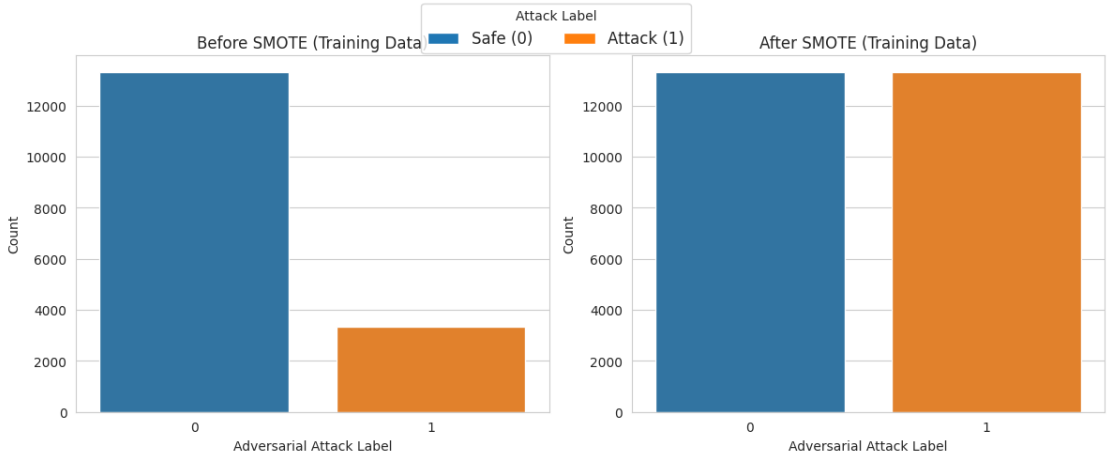
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20789 entries, 0 to 20788
Data columns (total 9 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Operating_Setting_2                20789 non-null  float64
 1   Corrected_Core_Speed_RPM           20789 non-null  float64
 2   Core_Speed_Variation               20789 non-null  float64
 3   Fuel_Air_Ratio                     20789 non-null  float64
 4   High_Pressure_Turbine_Bleed_Flow   20789 non-null  float64
 5   High_Pressure_Turbine_Bleed_Status 20789 non-null  float64
 6   Airflow_at_Station_31              20789 non-null  float64
 7   Remaining_Useful_Life_Status       20789 non-null  int64
 8   Adversarial_Attack_Label           20789 non-null  int64
dtypes: float64(7), int64(2)
memory usage: 1.4 MB
```

3) Test Train Split

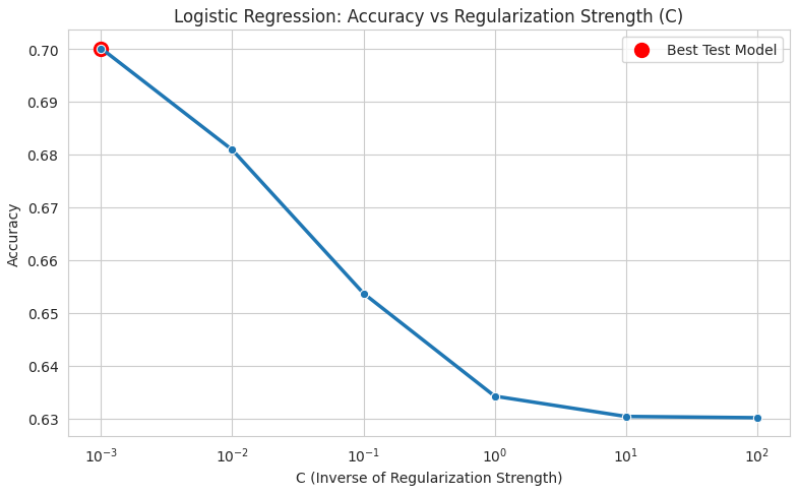
```
# Define features & target
X = df.drop(columns=['Adversarial_Attack_Label','High_Pressure_Turbine_Bleed_Flow','Fuel_Air_Ratio','Operating_Setting_2'])
y = df['Adversarial_Attack_Label'].astype(int)

# Split into train and test sets
SEED = 42
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.20,
    stratify=y,
    random_state=SEED
)
```

4) Applied SMOTE as dataset is imbalanced:

```
# Apply SMOTE
smote = SMOTE(random_state=SEED, k_neighbors=5)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```



5) Logistic regression plot:

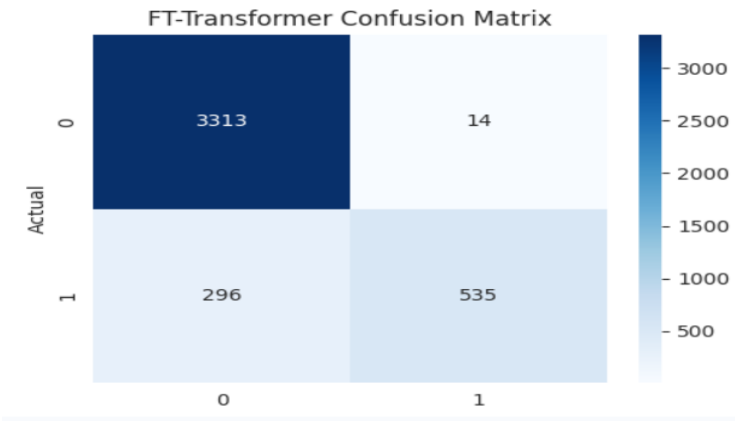


6) FT Transformer:

```
class FTTransformer(nn.Module):
    def __init__(self, num_features, d_token=128, n_heads=8, n_layers=4, ff_mult=4, dropout=0.2):
        super().__init__()
        self.feature_embed = nn.Linear(1, d_token)
        self.col_emb = nn.Parameter(torch.randn(num_features, d_token))
        self.cls_token = nn.Parameter(torch.randn(1, 1, d_token))

        encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_token,
            nhead=n_heads,
            dim_feedforward=d_token * ff_mult,
            activation='gelu',
            dropout=dropout,
            batch_first=True,
            norm_first=True
        )
```

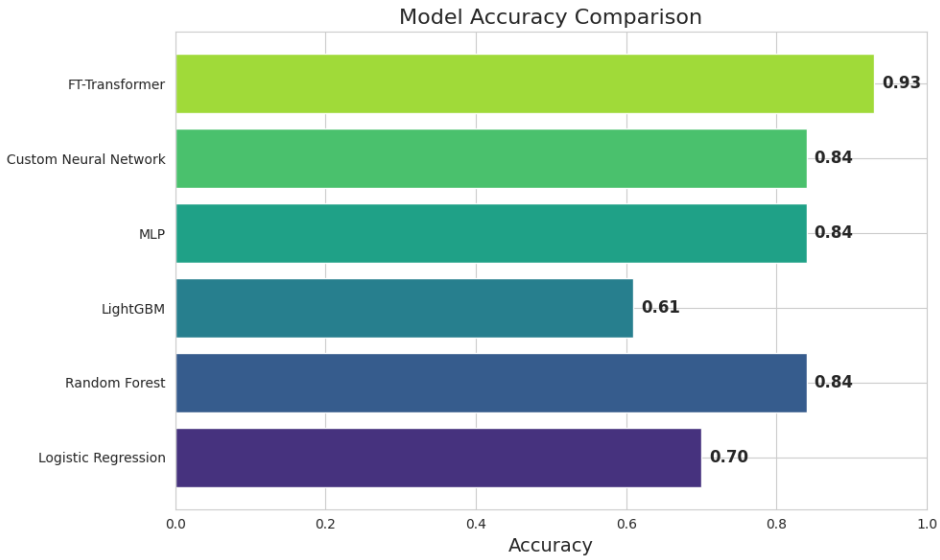
7) Classification report of FT Transformer:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      3327
           1       0.97      0.64      0.78       831

    accuracy                           0.93      4158
   macro avg       0.95      0.82      0.87      4158
weighted avg       0.93      0.93      0.92      4158
```



8) Model comparison:

The model comparison figure shows the variation in accuracy of all the six implemented models. Conventional machine learning models have poor performance with the weakest performance of Logistic Regression and LightGBM. The deep neural networks and Random Forest have slightly better accuracy but are not reliable in identifying the adversarial cases. The FT-Transformer is superior in the comparison and it has a definite separation with all baselines since it has the highest accuracy and most consistent detection behaviour across classes. The figure is a visual representation of how attention-based architectures are more robust to adversarial perturbations.



7. Conclusion:

The paper illustrates that standard deep learning models, as well as traditional machine learning, are capable of identifying general trends in engine sensor data, but they are weak against subtle adversarial perturbations. Conversely, the FT-Transformer is always higher in accuracy, recall, and F1-scores than any baseline models, as its attention-based feature interaction modelling makes it successful. The fact that it is able to dynamically capture irregular relationships makes it much more effective in discovering manipulated inputs. On the whole, the study validates that the transformer-based architecture is a more stable and dependable approach to detect adversarial attacks in sensor-based predictive maintenance systems.

References:

1. **https://numpy.org/doc/**
2. **https://scikit-learn.org/stable/**
3. **https://pytorch.org/**
4. **https://lightgbm.readthedocs.io/en/stable/**
5. **https://www.tensorflow.org/**
6. **https://pandas.pydata.org/docs/**